

1) Also, more details needed for the following:

Regarding the two advantages of UID:

1. What do you mean by 'saved space' here?
2. How is this saved space improves the throughputs.

Pl. give more explanation.

In the next paragraph, you have mentioned application-layer deep packet inspection can be done. Should we need to mention this part as future work?

Answer:

To identify the flow from the received data on split proxy, the join-proxy has to attach the flow information with the data. In the traditional network, the routing path is not fixed and the join-proxy doesn't have the global network information. The join-proxy can't decide which split-proxy will split the flow. So, the attached flow information of a TCP flow requires (IP source+IP destination+TCP source port+ TCP destination port)=4+4+2+2=12 byte to different with other TCP flow. However, the UID only requires 2 bytes. The TCP flow detail information is synchronized on join-proxy and split-proxy by centralized SDN controller. The UID saves 10 bytes. It improves the goodput for the joined long flow sending data of multiple short flows. For example, 10 short flows sending 1 byte each time. The joined long flow with UID+length+data is $(2+2+1)*10=50$ bytes. Comparing to $(12+2+1)*10=150$ bytes, the joined long flow will have better goodput.

Now the controller is already the bottleneck in the industry. We can do DPI on controller. But if the traffic is huge, it's better to run DPI as a NFV in the future.

2) What do you mean by the following phrase:

"4) When a ACK-join is released by ACK-split, and received on switch 1....."

Answer:

In the 3), when the ACK-split arrives at split-proxy, the split-proxy releases the ACK-join. In the 4), when the ACK-join arrives at join-proxy, join-proxy releases the ACK-client. Basically, I want to describe the process that server release the ack to the split-proxy, split-proxy release the ack to join-proxy, then the join-proxy release the ack to the client.

3) Explain what is an increaseACKValue function? What is the significant purpose of this function?

Answer:

For example, client A send 1000 byte to the join-proxy. Join-proxy has to send 1000+4 to the split-proxy. Then the split-proxy will send 1000 byte to the server.

The ACK from server to split-proxy is 1000,

The ACK from split-proxy to join-proxy is 1004,

The ACK from join-proxy to split-proxy is 1000,

The split-proxy receive ACK 1000 from server. But it requires 1004 to release the ACK to the join-proxy. "increaseACKValue" is to handle the extra 4 from 1000->1004 in the split-proxy. On the join-proxy it has a function opposite to "increaseACKValue" to handle the 1004->1000.

- 4) Suddenly, from nowhere, you mention delayed ACK, and combining several ACK responses, which is not clear. There seems to be a gap from previous paragraph and this paragraph.

Answer:

I think the delayed ACK should be ACK-join at here. What I want to describe is the system should aggregate ACK-split from different flows into one ACK-join.

- 5) Also, the Linked-ACK state diagram explanation is incomplete.
You need to introduce by explaining the traditional TCP state diagram.
The linked state-diagram PROXY is misspelled as PORXY, and FIN_WAIT_1 is not shown in full.

The following is the describe of the traditional TCP state diagram. I find it from TCP RFC from page 21 in this link: <https://tools.ietf.org/html/rfc793>

A connection progresses through a series of states during its lifetime. The states are: LISTEN, SYN-SENT, SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT, and the fictional state CLOSED. CLOSED is fictional because it represents the state when there is no TCB, and therefore, no connection. Briefly the meanings of the states are:

LISTEN - represents waiting for a connection request from any remote TCP and port.

SYN-SENT - represents waiting for a matching connection request after having sent a connection request.

SYN-RECEIVED - represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.

ESTABLISHED - represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase

of the connection.

FIN-WAIT-1 - represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.

FIN-WAIT-2 - represents waiting for a connection termination request from the remote TCP.

CLOSE-WAIT - represents waiting for a connection termination request from the local user.

CLOSING - represents waiting for a connection termination request acknowledgment from the remote TCP.

LAST-ACK - represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).

[Page 21]

Transmission Control Protocol
Functional Specification

September 1981

TIME-WAIT - represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request.

CLOSED - represents no connection state at all.

A TCP connection progresses from one state to another in response to events. The events are the user calls, OPEN, SEND, RECEIVE, CLOSE, ABORT, and STATUS; the incoming segments, particularly those containing the SYN, ACK, RST and FIN flags; and timeouts.

The state diagram in figure 6 illustrates only state changes, together with the causing events and resulting actions, but addresses neither error conditions nor actions which are not connected with state changes. In a later section, more detail is offered with respect to the reaction of the TCP to events.